Investigating the Effect of Methods of Trade-Off Between Exploration and Exploitation on the Return in Reinforcement Learning Algorithms

Avantika Chopra

May 2023

1 Introduction

As Alpha Go (a computer program that plays the board game Go) made its 37th move against Lee Sedol, the popularity of reinforcement learning algorithms grew. While playing against the program, Lee Sedol said "This move (move 37) made me think about Go in a new light. What does creativity mean in Go?", setting reinforcement learning apart from supervised and unsupervised learning. Such algorithms primarily aim to maximize rewards as opposed to finding structure.

Reinforcement learning algorithm agents learn by interacting with their environment in order to determine the immediate action with the highest yielding reward and actions over an extended period of time that would yield the highest cumulative reward. An agent's knowledge is always limited to what it has already explored in the past and can solely choose to exploit that knowledge to obtain its reward – greedy selection. It can also explore new states to improve its actions and gain better rewards. However, both cannot occur together. This is known as the dilemma between exploration and exploitation.

The objective of this paper is to explore the extent to which the methods ϵ-Greedy and Boltzmann Selection) of trade-off between exploration and exploitation effect the cumulative reward in on-learning and off-learning reinforcement learning algorithms (SARSA and Q-learning).

Such an investigation can yield better results for reinforcement learning algorithms applications such as advertisement platforms. Simply exploiting the agent's knowledge would show users advertisements that they are most likely to click on and thereby drive short-term revenue. Whereas exploration would allow the agent to learn more about users' preferences and find more relevant advertisements in the future.

2 Literature Review

2.1 Reinforcement Learning

Reinforcement learning is a field of machine learning wherein an agent must learn how to optimize the reward signal through exploration. Actions made by the agent affect not only the immediate reward, but also future actions and rewards associated with them.

An *agent* carries out *actions* in an *environment* that result in a new *state*. Environments are the situations with which agents interact. *Policies* define an agent's behaviour and may be discreet-time stochastic processes. The *reward signal* defines the immediate goal of the reinforcement learning problem. With each action and change in state, the agent receives a reward which it must try to maximize over an extended period of time. Positive reinforcement in the form of positive integers increases an agent's tendency to perform an action. Negative Reinforcement however, decreases the strength of the behaviour. The *value function* defines the long-term goal of the reinforcement learning problem. It defines the cumulative reward an agent can expect to receive by moving to a certain state.

The Markov Decision Process states that the future is independent of the past actions, rewards and states. It can be defined by the following tuple of elements:

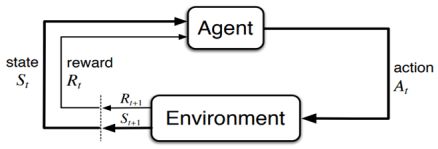

Figure 1: Adapted from Sutton and Barto

Agents interact with the environment at discreet time steps and makes an action A_t from its current state S_t where $A(S_t)$ are the available actions. Upon selecting the action, the agent receives reward R_{t+1} at the state S_{t+1}. The policy π_t maps the probability of selecting the possible action to states. Thus, $\pi_t(a|s)$ denotes the probability that $A_t = a$ when $S_t = s$. The value $Q_t(a)$ gives the estimate of an action's value at time t. It can be updated by Bellman's equation.

$$NewEstimate \leftarrow OldEstimate + StepSize[Target - OldEstimate]$$

The step-size parameter α is configurable between the values 0 and 1 and describes the changes between each time step. In layman's terms, it controls how fast do Q-function forgets old values in favour of new experiences. "Unfortunately, we cannot analytically calculate the optimal learning rate for a given model on a given dataset. Instead, a good (or good enough) learning rate must be discovered via trial and error." For the purpose of this investigation, the

value of this parameter was arbitrarily selected. The return G_t is the sum of the rewards

$$G_t = \sum_{k=0}^{\infty} R_{k+1}$$

When maximizing the expected return, the concept of discounting is used. The discount rate γ is a parameter that indicates that a reward received k time steps in the future is worth $\gamma(k-1)$ times what its immediate worth would be. "As in MDP's, the discount factor can be thought of as the probability that the game will be allowed to continue after the current move. This is because, in many games, it is best to postpone risky actions indefinitely. The discount factor has the desirable effect of goading the players into trying to win sooner rather than later." For the purpose of this investigation, the value of this parameter was arbitrarily selected.

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{k+1}$$

2.2 SARSA and Q-Learning

State Action Reward State Action or SARSA is an on-policy reinforcement learning algorithm. It differs from Q-learning in its updating strategy for the value function. The updating strategy target is the sum of the next state's reward and the discount factor multiplied by the value of the next chosen state-action pair.

$$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$$

Q-Learning is an off-policy reinforcement learning algorithm. In its updating strategy for the value function depends on the optimal value of the next state-action pair. The updating strategy target is the sum of the next chosen state's reward and the discount factor multiplied by the maximum value from the next 4 possible state-action pairs, not necessarily the chosen state.

$$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma max_{\alpha} Q(S', A) - Q(S, A)]$$

2.3 Exploration and Exploitation

An agent must interact with its environment multiple times in order to learn the actions that maximize its reward. Exploitation refers to when an agent takes advantage of its current knowledge on a set of estimated state-action pair values in order to obtain favourable long-term rewards. If the agent only exploits its existing knowledge, it will converge and result in the same local optimum reward.

On the other hand, exploration is when an agents samples actions from a set of action in order to obtain better rewards. However, exploration is costly

in terms of resources, time and opportunity as the agent may risk obtaining a lower cumulative reward. Thus, it is important to explore the optimal trade-off between exploration and exploitation.

2.4 ϵ-Greedy Selection and Boltzmann Selection

There are numerous strategies used to balance exploration and exploitation and the two being explored in this paper are: ϵ-Greedy Selection and Boltzmann Selection.

ϵ-Greedy Selection is one of the simplest methods where an agent randomly selects an action with probability ϵ – exploration –and the action with the maximum value – exploitation – with probability 1-ϵ.

Boltzmann Selection is where the probability of selecting an exploratory action depends on how the state-action values compare to each other . The probability of how optimal action a at state s is given by the following equation.

$$p = \frac{e^{Q(s,a)}}{\sum_a e^{\frac{Q(s,a)}{\tau}}}$$

The temperature τ is a parameter that determines the randomness in the selection of an action. An increasing temperature τ, increases the rate of exploration.

3 Methodology

3.1 Variables

3.1.1 Independent Variables

ϵ **Value** in ϵ-Greedy Selection will vary the probability with which an exploratory action will be selected. The value of ϵ will be stored in the variable epsilon. Epsilon values lie within the range of $0 \leq \epsilon \leq 1$. Values of ϵ less than 0.5 have been selected at 0.1 intervals to determine the effect of no exploration and an equal probability of exploration and exploitation.

Temperature in Boltzmann Selection will vary the rate of exploration. The value of τ will be stored in the variable temperature. The temperature values are $\tau \geq 0$. As $\tau \to \infty$, the agent tends to exploration moves and as $\tau \to 0$, it tends of exploitation. Larger values of τ were selected in comparison to ϵ, as there were very few observable changes in such small increments and thus corresponding values to ϵ were taken to the power of 100.

3.1.2 Dependent Variables

Return or the cumulative reward received when the agent reaches the environment goal will be stored in the variable sum. A higher return indicates

expediency in an algorithm as the reward function provides negative reinforcement of -1 for each time step. An increasing return indicated better performance by the agent.

Total Regret is the sum of opportunity loss for each time step. Minimal total regret corresponds with maximum return and is another measure of expediency in an algorithm. It can be found by the sum of the maximum state-action pair value subtracted by the chosen state-action pair value for each time step. However, as the Total regret is simply the return(episode) function reflected over the x axis, it will not be calculated.

Length of each Episode indicates how many time steps the agent took to reach its goal. It is important to note that although the reward function does give the agent a reward of -1 for each time step, it also gives a -100 reward if it falls into the cliff and thus, the return cannot solely be used as a measure of expediency for the algorithm. A decreasing length of steps in each episode indicates better performance of the agent.

Q-matrix shows the value of a state-action pair. By analysing the maximum value for each state-action pair and comparing it with the optimum state-action pairs, the expediency can be analysed. This measures how well the agent would exploit its knowledge.

3.1.3 Control Variables

Variable	Value	Notes
Step-size Parameter α	0.8	
Discount Rate γ	0.75	
Total Episodes	500	The return converges after 500 episodes
Maximum Steps	100	In consideration of time and space limitations
Environment Size	$4x12$	Default size of Cliff Walking-v0 environment
Reward Function	-100 at cliff, -1 for each time step, $+100$ for goal	Modified function of Cliff Walking-v0 environment
Trials	10	To reduce fluctuations in the return and steps in each episode

3.2 Environment

Gym open-source python library will be used to create the Cliff Walking Environment Version 1 of default size $4x12$. The agent must navigate the grid world from Start(S) at [3][0] to Goal(G) at [3][11]. Each time step provides negative reinforcement of -1 and stepping into the cliff at $[3][1 \dots 10]$ provides a -100 reward. The Goal provide positive reinforcement (Cliff Walking) of $+100$. The

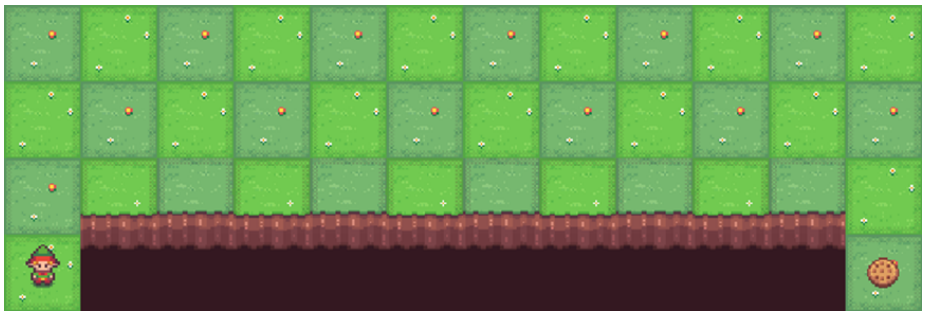

Figure 2: Cliff Walking

actions the agent can make are 1-element vectors: 0 (Up), 1 (Right), 2 (Down), 3(Left).

3.3 Updating Strategies

3.3.1 SARSA

$//S, S' \rightarrow$ states
$//A, A' \rightarrow$ actions
$//Q \rightarrow$ state-action value
$\alpha, \gamma \rightarrow$ step-size parameter, discount factor
Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
    Initialize S
    Choose A from S using policy derived from Q (ϵ-greedy or Boltzmann)
    Repeat (for each step of episode):
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q
        $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma max_{\alpha} Q(S', A) - Q(S, A)]$
        $S \leftarrow S', A \leftarrow A'$

    Until S is terminal

3.3.2 Q-Learning

$//S, S' \rightarrow$ states
$//A, A' \rightarrow$ actions
$//Q \rightarrow$ state-action value
$\alpha, \gamma \rightarrow$ step-size parameter, discount factor
Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode): Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q

Take action A, observe R, S'
$Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma max_\alpha Q(S',A) - Q(S,A)]$
$S \leftarrow S'$
Until S is terminal

3.4 Selecting an Action

3.4.1 ϵ-Greedy Selection

$//\epsilon \rightarrow$ epsilon
$//S \rightarrow$ state
$//A \rightarrow$ action
$//as \rightarrow$ action space
Choose Action (S)
if random sample from uniform distribution $< \epsilon$ then
$A \leftarrow$ Sample from as
else
$A \leftarrow max_a Q(S,a)$

3.4.2 Boltzmann Selection

$//\tau \rightarrow$ temperature
$//S \rightarrow$ state
$//A \rightarrow$ action
$//as \rightarrow$ action space
Choose Action (S)
Repeat i for range (as): Boltzmann Probability $\leftarrow \frac{e^{\frac{Q(s,i)}{\tau}}}{\sum_{a=0}^{as} e^{\frac{Q(s,a)}{\tau}}}$
$A \leftarrow$ Non-uniform random sample of as based on Boltzmann Probability

4 Experimental Results

4.1 $\epsilon = 0.1$, $\tau = 10$